A- Orientações para instalação e configuração de ambiente de testes

A.1- Introdução

Neste apêndice, sugerimos um exemplo de ambiente de teste de Caixa Preta em ativos de TIC.

A.1.1 Rede

Para configuração do ambiente, algumas premissas básicas:

1. Conectividade com a Internet
2. Plano de endereçamento IP interno
3. Equipamento para prover conectividade com a Internet e o plano de endereçamento IP
4. Servidor para instalação do Ambiente de Monitoração
5. Servidor para instalação do Ambiente de Virtualização

A definição de plano de endereçamento IP ajudará na identificação dos ativos de TIC que estarão sujeitos a teste e dos que não estarão.

O plano de endereçamento inicial, para rede interna, sugerimos 192.168.1.0/24. Que o equipamento a prover conectividade com a Internet tenha o IP 192.168.1.1/24; o servidor para instalação do Ambiente de monitoração tenha um fixo e como sugestão 192.168.1.2/24; ; o servidor para instalação do Ambiente de virtualização tenha um fixo e como sugestão 192.168.1.3/24; qualquer outro equipamento que não esteja em teste e

Rede Interna - 192.168.1.0/24		
Nome do ativo de TIC	IP	MAC
Roteador/Modem com Wifi	192.168.1.1	52:54:00:4d:6b:a0
Monitoração	192.168.1.2	52:54:00:4d:6b:b0
Virtualização	192.168.1.3	52:54:00:4d:6b:c0
	...	
Faixa DHCP	192.168.1.50 à 192.168.1.254	

Tabela 4 – Exemplo para documentação de Rede

faça parte de ambiente também tenha um IP fixo e; todos os outros ativos de TIC que estejam em Teste estejam em uma faixa para DHCP dentro da rede definida.

Ainda sobre endereçamento e identificação dos ativos de TIC, sugerimos, a tabela abaixo e que se for possível, ocorra a identificação dos endereços de MAC de todos os ativos de TIC, pois, como será utilizada uma faixa DHCP para os ativos em teste, os mesmos poderão mudar de IP dentro desta faixa. Via de regra, os equipamentos que proveem conectividade têm esta informação.

O endereçamento com a Internet deverá ser fornecido e configurado previamente no equipamento que proverá a conectividade com a Internet. Este deverá estar previamente configurado para o acesso a Internet e com um *firewall* para proteção.

Ainda sobre o endereçamento para rede interna (192.168.1.0/24), este poderá estar junto do equipamento que proverá a conectividade com a Internet. Este equipamento deve ter a capacidade de fornecer um serviço DHCP e estar configurado conforme a sugestão da tabela acima.

Abaixo equipamentos que testamos e que tem as características que necessitamos para conectividade com a Internet:

1. Modem e Roteador Wireless Technicolor TG588V V2 ADSL VDSL
2. Modem e Roteador Wireless Technicolor TD5336
3. Roteador Wireless C3-TECH W-R2000nL
4. Switch Cisco IOS Software, Catalyst 4500 L3
5. Roteador e/ou Modem com Wireless e suporte ao sistema operacional OpenWRT 18.0X

O servidor para instalação do Ambiente de Monitoração, que sugerimos ser configurado na rede interna com IP 192.168.1.2, deve ser mensurado de acordo com o volume de tráfego e dados que devem ser guardados para análise dos Teste de Caixa Preta em ativos de TIC. A configuração mínima para o mesmo sugerida é :

1. 8GB de memória RAM
2. Armazenamento de 500 GB
3. Processador com suporte a no mínimo 04 CPUs
4. Placa de rede com suporte a *FastEthernet* 100Mbps
5. Sistema Operacional Linux Ubuntu 16 ou CentOS 7

Lembrando que caso seja utilizada a opção de *sniffer*, a partir do ambiente de monitoração, é necessária a verificação de suporte e velocidade da placa de rede do servidor.

Ettercap

O pacote Ettercap é instalado a partir *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará capturar o tráfego de rede caso seja utilizada a técnica *sniffing*. Além disso, este pacote deverá ser executado no mesmo servidor que estará o Suricata IDS e o Evebox.

Para configurar qualquer item do Ettercap, este deve ser configurado no diretório */etc/ettercap* e edite os arquivos do diretório. Estes são os seguintes:

1. etter.conf - Configuração do Ettercap
2. etter.dns - Configuração do plug-in dns_spoof do Ettercap
3. etter.mdns - Configuração do plug-in mdns_spoof do Ettercap
4. etter.nbns - Configuração do plug-in nbns_spoof do Ettercap

Edição de configuração do Ettercap

```
# cd /etc/ettercap
# vi etter.conf
```

Diferente dos outros serviços a execução do Ettercap deve ser manual. Para isso, será iniciado um *screen* e a partir do mesmo será feita a execução. Para exemplos de execução, verifique em *https://github.com/carlos-teles/etsg/blob/master/start-ettercap.sh* . Execute o comando abaixo, como root, onde 192.168.1.1. é o IP de saída para Internet e *gateway* da rede para capturar o tráfego de rede:

Executar Ettercap dentro de screen

```
# screen -S ettercap
# /usr/bin/ettercap -T -M arp:remote /192.168.1.1// ///
```

Após a inicialização do Ettercap deveremos ter a saída abaixo em execução (Figura 30).

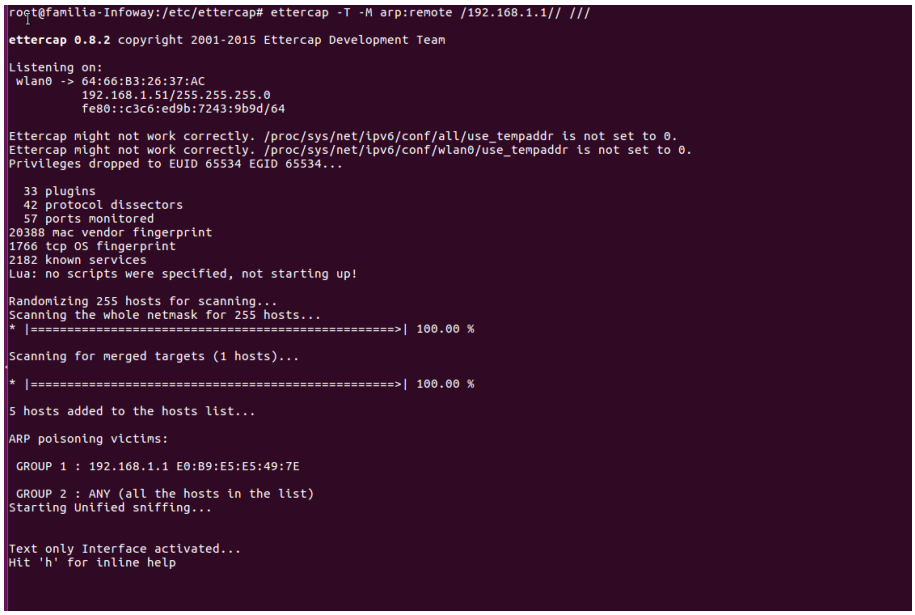

Figura 30 – Tela execução do Ettercap

Espelhamento de portas

O Espelhamento de portas é uma técnica para captura de tráfego de rede. Neste caso, temos duas interfaces de rede. Uma por onde passa todo o tráfego de rede

(entrada e saída) e outra que receberá uma cópia. Esta configuração foi executada em um Switch Cisco IOS Software, Catalyst 4500 L3. As interfaces selecionadas são GigabitEthernet6/21 (Gi6/21) e GigabitEthernet4/39 (Gi4/39). A interface Gi6/21 receberá uma cópia de todo o tráfego da interface Gi4/39, que é por onde passa todo tráfego de entrada e saída que determinamos. Segue abaixo a configuração já aplicada:

Exemplo de configuração de Espelhamento de portas

```
swcore01.rjo#sh run | i monitor session
monitor session 3 source interface Gi4/39
monitor session 3 filter packet-type good rx
monitor session 3 destination interface Gi6/21
```

Para verificar se a configuração de espelhamento de portas ficou correta, execute o comando abaixo:

Verificação de Configuração de Espelhamento de portas

```
swcore01.rjo#show monitor session all
Session 3
---------
Type                   : Local Session
Source Ports           :
Both               : Gi4/39
Destination Ports      : Gi6/21
Encapsulation      : Native
Ingress      : Disabled
Learning : Disabled
Filter Pkt Type        :
RX Only       : Good
swcore01.rjo#
```

A.1.2 Monitoração

O Ambiente de Monitoração deve ser instalado em um Servidor com sistema Operacional Linux. Utilizamos como base as versões de 16.XX do Ubuntu e também as versões de CentOS 7. Em ambos instalamos os pacotes e funcionaram corretamente.

Abaixo algumas das configurações utilizadas.

Hardware do Servidor de Monitoração					
#	CPU	Memória	Armazenamento	Rede	Sistema Operacional
1	Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz	8GB	148GB	Realtek RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 02)	Ubuntu 16.04.6 LTS
2	Intel(R) Xeon(R) CPU 5140 @2.33GHz	8GB	126GB	Intel Corporation 82571EB Gigabit Ethernet Controller (rev 06)	Kali GNU/Linux Rolling
3	Intel Pentium 4 @3.4GHz	6GB	500GB	Braodcom NetXtreme BCM5721 Gigabit Ethernet	Linux Mint 18.3

Para facilitar a instalação, desenvolvemos um script que testar a versão de sistema operacional e que instala os pacotes já citamos acima. Este script encontra em https://github.com/carlos-teles/etsg/ e possui o nome ”install-only-etsg.sh”.

Resumidamente, os pacotes a serem instalados são:

1. Softwares básicos caso não estejam instalados como *OpenSSH*, Servidor DHCP e atualizações de Python
2. Suricata IDS
3. EVEBOX
4. Grafana
5. Influxdb e Telegraf
6. Suricata-update (atualização de regras do Suricata IDS)

Uma vez que todos os pacotes estejam instalados é necessário que sejam verificados os estados dos processos que devem estar em execução.

Influxdb e Telegraf

Os pacotes Influxdb e Telegraf são instalados a partir *script* acima (Seção A.1.2), mas não são habilitados para iniciar automaticamente, pois, os mesmos devem ser configurados previamente. Estes pacotes ajudarão a apresentar informações de Desempenho.

O pacote Telegraf, caso queira configurar algum item, este deve ser configurado no diretório */etc/telegraf* e edite o arquivo *telegraf.conf*.

Edição de configuração do Telegraf

```
# cd /etc/telegraf
# vi telegraf.conf
```

Em nosso caso é necessário alterar as configurações, pois, precisamos que todos os dados de tráfego de rede sejam coletados. Assim, sugerimos remover os comentários das linhas abaixo, ficando da seguinte forma:

Edição de configuração do Telegraf

```
# # Read metrics about network interface usage
[[inputs.net]]
#   ## By default, telegraf gathers stats from any up interface (excluding loopback)
#   ## Setting interfaces will tell it to gather these explicit interfaces,
#   ## regardless of status.
#   ##
interfaces = ["eth0"]
#   ##
#   ## On linux systems telegraf also collects protocol stats.
#   ## Setting ignore_protocol_stats to true will skip reporting of protocol metrics.
#   ##
#   # ignore_protocol_stats = false
#   ##
```

É importante lembrar que cada distribuição linux pode apresentar nomes diferentes de interfaces. Em nosso caso, nossa interface é a *eth0*. Podemos ter diversas e que gostaríamos que fossem monitoradas. Desta forma, caso tenhamos em nosso sistema *eth0*, *eth1* e *eth2*, a linha acima ficaria:

Edição de configuração do Telegraf

```
# # Read metrics about network interface usage
[[inputs.net]]
#   ## By default, telegraf gathers stats from any up interface (excluding loopback)
#   ## Setting interfaces will tell it to gather these explicit interfaces,
#   ## regardless of status.
#   ##
interfaces = ["eth0","eth1","eth2"]
#   ##
#   ## On linux systems telegraf also collects protocol stats.
#   ## Setting ignore_protocol_stats to true will skip reporting of protocol metrics.
#   ##
#   # ignore_protocol_stats = false
#   ##
```

Caso queira configurar algum item do Influxdb, este deve ser configurado no diretório */etc/influxdb* e edite o arquivo *influxdb.conf*.

Edição de configuração do Influxdb

```
# cd /etc/influxdb
# vi influxdb.conf
```

Neste caso, para efeito de backup, sugerimos remover o comentário da linha, ficando da seguinte forma:

Edição de configuração do Telegraf

```
bind-address = "127.0.0.1:8088"
```

Assim, agora, poderemos continuar com as verificações. Os pacotes não vêm para serem executados automaticamente. Verificaremos então seus status.

E devem estar com as seguintes saídas:

Verificar status do Telegraf

```
# systemctl status telegraf.service
 telegraf.service - The plugin-driven server agent for reporting metrics into I
Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset:
Active: inactive (dead) since Dom 2019-05-05 01:02:09 -03; 9s ago
Docs: https://github.com/influxdata/telegraf
Process: 1274 ExecStart=/usr/bin/telegraf -config /etc/telegraf/telegraf.conf
Main PID: 1274 (code=exited, status=0/SUCCESS)
```

Verificar status do Influxdb

```
# systemctl status influxd.service
 influxdb.service - InfluxDB is an open-source, distributed, time series databa
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset:
Active: inactive (dead) since Dom 2019-05-05 01:02:14 -03; 1min 5s ago
Docs: https://docs.influxdata.com/influxdb/
Process: 1396 ExecStart=/usr/bin/influxd -config /etc/influxdb/influxdb.conf
Main PID: 1396 (code=exited, status=0/SUCCESS)
```

Verificamos assim que ambos os serviços, telegraf e influxdb, estão inativos. Para iniciar e verificar o status destes, execute os comandos abaixo:

Iniciar e Verificar status do Telegraf

```
# systemctl start telegraf.service
# systemctl status telegraf.service
 telegraf.service - The plugin-driven server agent for reporting metrics into I
Loaded: loaded (/lib/systemd/system/telegraf.service; enabled; vendor preset:
Active: active (running) since Dom 2019-05-05 01:07:57 -03; 12s ago
Docs: https://github.com/influxdata/telegraf
Main PID: 9195 (telegraf)
CGroup: /system.slice/telegraf.service
9195 /usr/bin/telegraf -config /etc/telegraf/telegraf.conf -config-
```

Iniciar e Verificar status do Influxdb

```
# systemctl start influxd.service
# systemctl status influxd.service
 influxdb.service - InfluxDB is an open-source, distributed, time series databa
Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; vendor preset:
Active: active (running) since Dom 2019-05-05 01:08:04 -03; 1min 4s ago
Docs: https://docs.influxdata.com/influxdb/
Main PID: 9216 (influxd)
CGroup: /system.slice/influxdb.service
9216 /usr/bin/influxd -config /etc/influxdb/influxdb.conf
```

A partir deste momento, os serviços estão em execução. Para que eles possam iniciar automaticamente, ou seja, caso seja feito um *reboot* do sistema operacional, os seguintes comandos também devem ser executados:

Habilitar Influxdb e Telegraf na inicialização

```
# systemctl enable influxd.service
# systemctl enable telegraf.service
```

Grafana

Durante a instalação do Grafana, o mesmo é habilitado automaticamente. Para se certificar, como root, execute:

Verificar status do Grafana

```
# systemctl status grafana-server

 grafana-server.service - Grafana instance
Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; ven
Active: active (running) since Dom 2019-05-05 01:20:38 -03; 1s ago
Docs: http://docs.grafana.org
Main PID: 9416 (grafana-server)
CGroup: /system.slice/grafana-server.service
9416 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --p
```

Após esta verificação, devemos acessar pelo *browser* o servidor do grafana. Por padrão, este é executado na porta 3000. A URL para acesso é *http://192.168.1.2:3000/login* , caso você tenha utilizado o IP sugerido acima. Ao acessar a seguinte tela aparecerá.

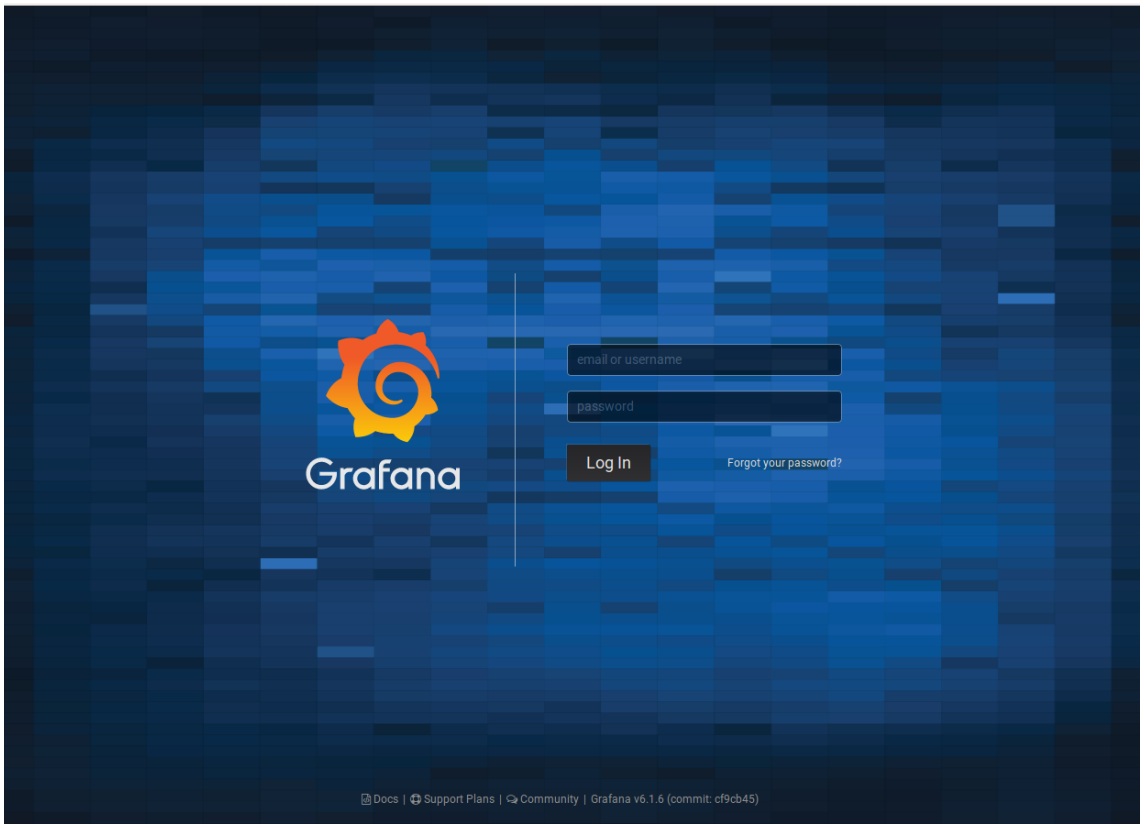

Figura 31 – Tela inicial do Grafana

Neste primeiro acesso o login e senha padrão do sistema são 'admin' e 'admin' respectivamente. Será solicitada a troca se senha após o primeiro login e a senha não pode ser 'admin'.

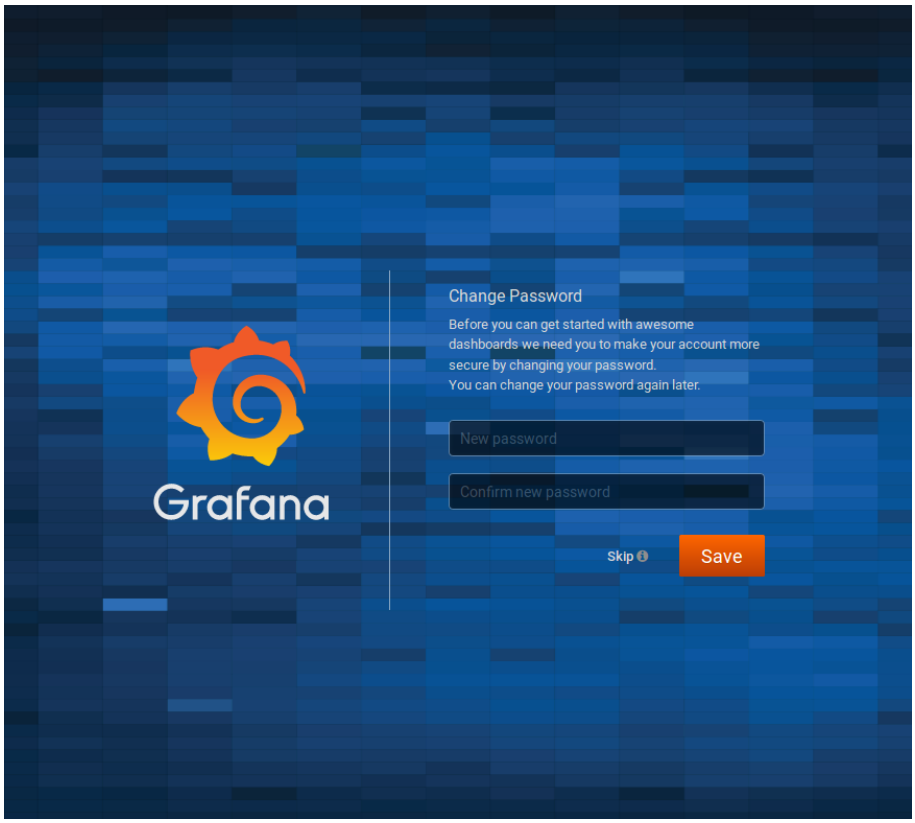

Figura 32 – Troca de senha inicial do Grafana

Após a troca de senha, a tela inicial do Grafana, vide figura 33.

Para mostrar os dados coletados pelo Telegraf e guardados no Influxdb é necessário acrescentar um *data source*. Clique em *Add data source*, que está selecionado na figura acima. Serão mostrados diversos *data sources* suportadas pelo Grafana. Em nosso caso utilizaremos o Influxdb. Clique no ícone relacionado conforme a figura 34.

Após clicar a tela abaixo (Figura 35) será carregada e deverá ser preenchida conforme mostrada.

Depois de preencher os campos, basta clicar no botão *Save & Test*, no rodapé da página (Figura 36). Caso tenha preenchido tudo corretamente a mensagem *Data source is working* aparecerá em verde.

Com o *data source* 'Influxdb' configurado e testado (Figura 37) é necessário acrescentar um *dashboard* para exibir as informações coletadas. Esta é uma das facilidades do Grafana, pois, é possível pesquisar no site do Grafana, em *https://grafana.com/dashboards*,

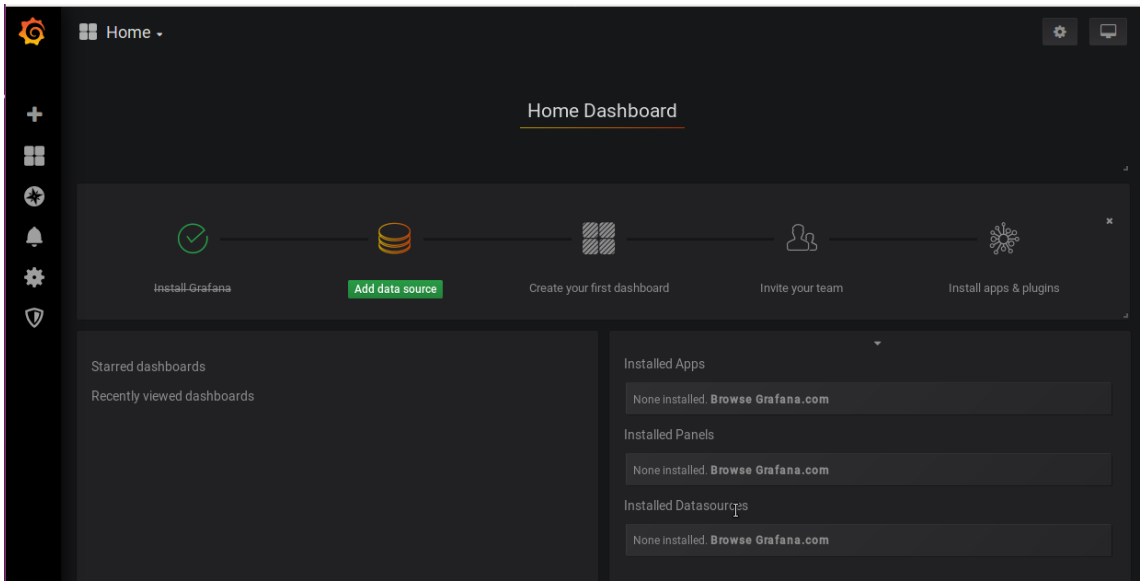

Figura 33 – Tela inicial do Grafana após login

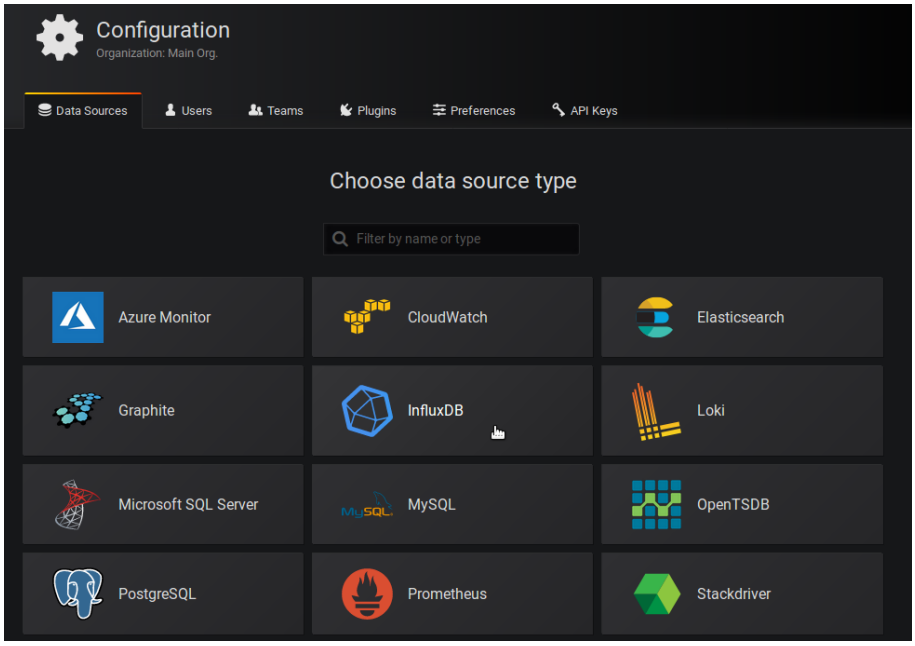

Figura 34 – Tela *Add data source* do Grafana

centenas de *dashboards* disponíveis e facilmente importados para o sistema. Sugerimos que seja importado para o Grafana o *dashboard* com o ID 928. Este possui uma documentação em *https://grafana.com/dashboards/928* .

Para importar no Grafana o *dashboard* com o ID 928 clique no ícone de 'mais', no canto superior esquerdo (Figura 38) e depois em *Import*. Em seguida a tela para importar será carregada (Figura 39).

Na tela carregada (Figura 39), clicar na caixa de texto com título *Grafana.com*

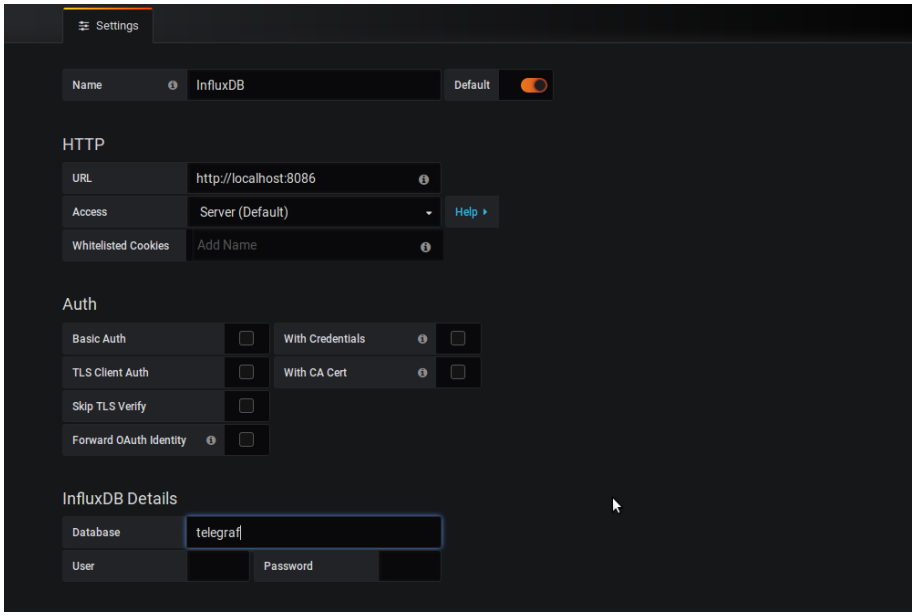

Figura 35 – Configurar *data source* do Grafana

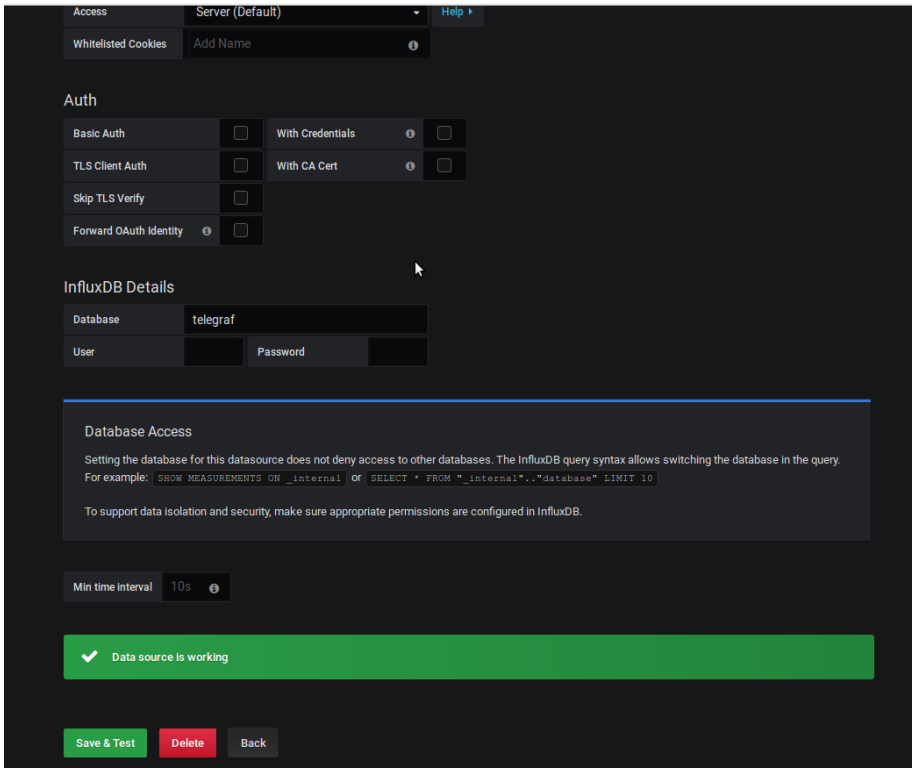

Figura 36 – Salvar configuração do *data source* do Grafana

Dashboard e digite o número 928. Depois clique no botão *Load* em azul para importar o *dashboard*. As configurações do *dashboard* são carregadas (Figura 40), bastando selecionar no campo *InfluxDB telegraf* o *data source* que criamos previamente com o nome de 'InfluxDB' e clicando no botão verde *Import*.

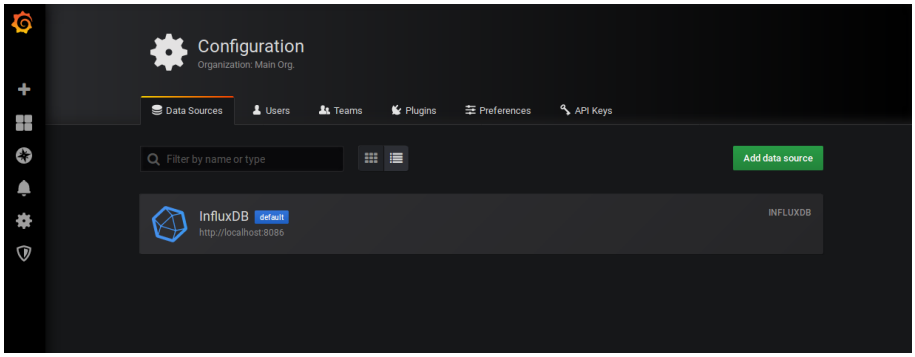

Figura 37 – Inflxudb *data source* do Grafana

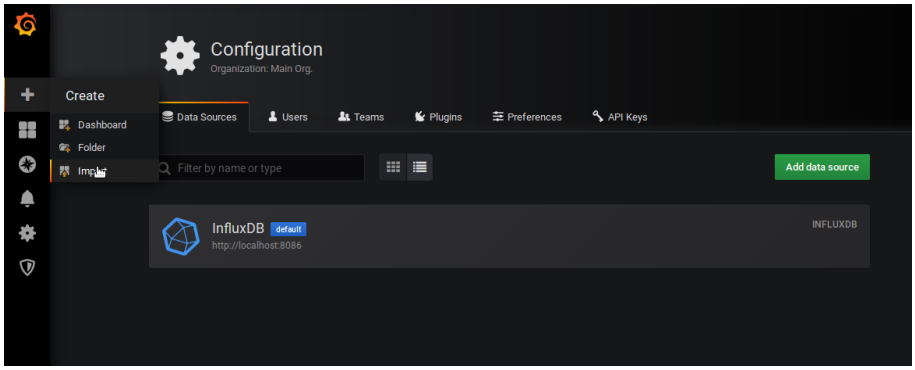

Figura 38 – Como Importar dashboard

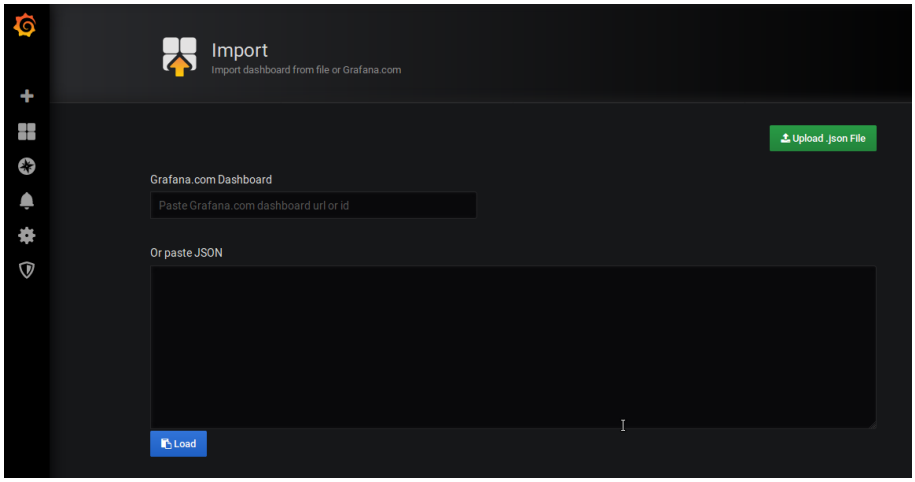

Figura 39 – Importar *dashboard*

Logo após clicar no botão *Import* o Grafana mostrará o *dahsboard* já com dados coletados. Abaixo algumas das informações carregadas.

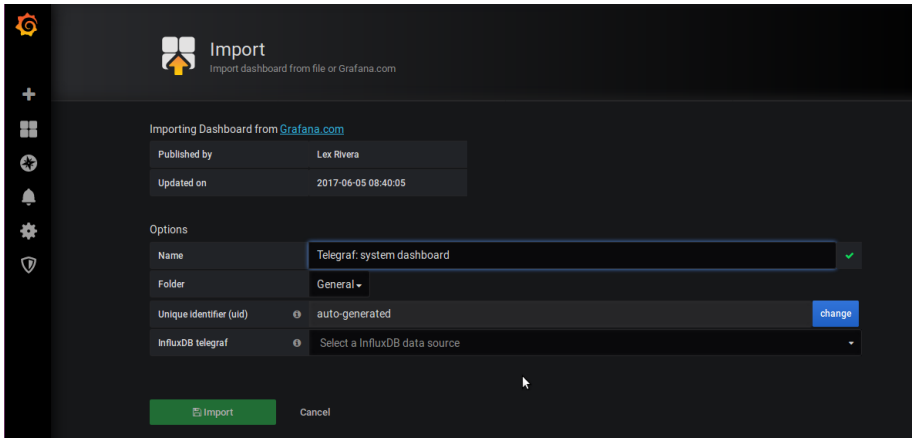

Figura 40 – Dados do *dashboard* 928 - 1

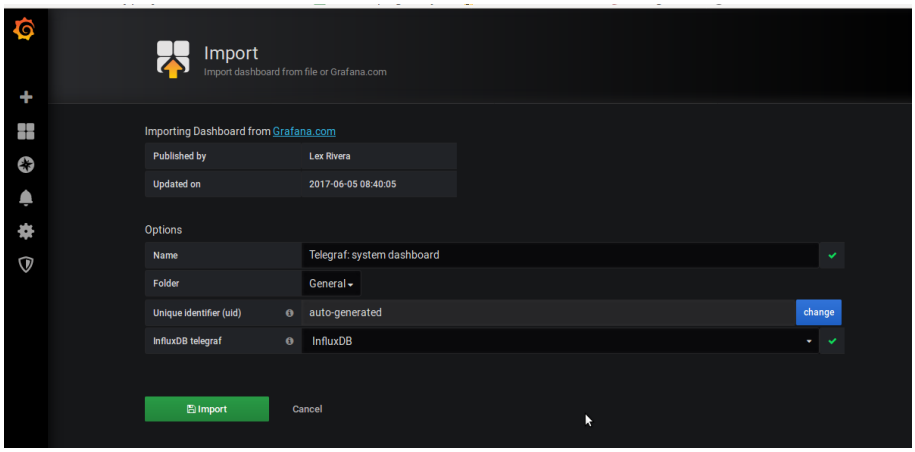

Figura 41 – Dados do *dashboard* 928 - 2

Suricata IDS

O pacote Suricata IDS é instalado a partir *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará a coletar informações de Segurança.

Para configurar qualquer item do Suricata, este deve ser configurado no diretório */etc/suricata* e edite o arquivo *suricata.yaml*.

Edição de configuração do Suricata

```
# cd /etc/suricata
# vi suricata.yaml
```

O pacote não vem para serem executados automaticamente. Verificaremos então seu status e deve estar com a seguinte saída:

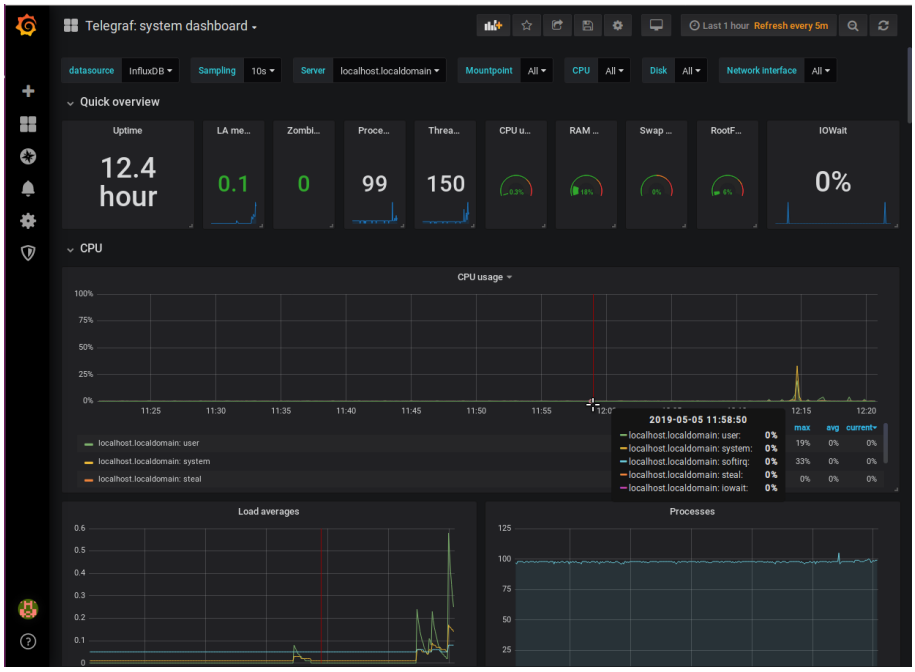

Figura 42 – *Dashboard* 928 - 01

Figura 43 – *Dashboard* 928 - 02

Verificar status do Suricata

```
# systemctl status suricata.service
 suricata.service - LSB: Next Generation IDS/IPS
Loaded: loaded (/etc/init.d/suricata; bad; vendor preset: enabled)
Active: inactive (dead)
Docs: man:systemd-sysv-generator(8)
```

Figura 44 – *Dashboard* 928 - 03

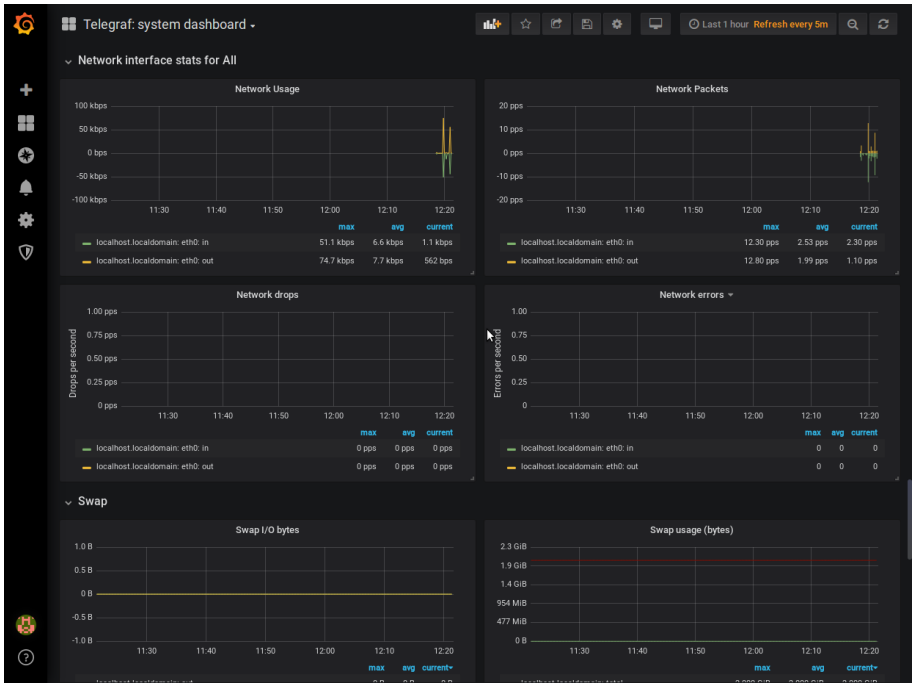

Figura 45 – *Dashboard* 928 - 04

Verificamos assim que o serviço do Suricata está inativo. Para iniciar e verificar o status destes, execute os comandos abaixo:

Iniciar e Verificar status do Suricata

```
# systemctl start suricata.service
# systemctl status suricata.service
 suricata.service - LSB: Next Generation IDS/IPS
Loaded: loaded (/etc/init.d/suricata; bad; vendor preset: enabled)
Active: active (running) since Seg 2019-05-06 00:19:22 -03; 1s ago
Docs: man:systemd-sysv-generator(8)
Process: 22588 ExecStart=/etc/init.d/suricata start (code=exited, status=0/SUC
CGroup: /system.slice/suricata.service
22597 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /v
```

A partir deste momento o serviço está em execução. Para que possa iniciar automaticamente, ou seja, caso seja feito um *reboot* do sistema operacional, o seguinte comando também deve ser executados:

Habilitar Suricata na inicialização

```
# systemctl enable suricata.service
```

Evebox

O pacote Evebox é instalado a partir *script* (Seção A.1.2), mas, também não é habilitado para iniciar automaticamente. Este pacote ajudará a mostrar as informações de Segurança coletadas.

Para configurar qualquer item do Evebox, este deve ser configurado no diretório */etc/evebox* e faça uma cópia do arquivo *evebox.yaml.example* para *evebox.yaml*.

Edição de configuração do Evebox

```
# cd /etc/evebox
# cp evebox.yaml.example evebox.yaml
# vi evebox.yaml
```

Diferente dos outros serviços, como não utilizaremos neste momento o Elasticsearch, pois, esta é uma abordagem leve para o ambiente, a execução do Evebox será manual. Para isso, será iniciado um *screen* e a partir do mesmo será feita a execução. Execute os comandos abaixo:

Executar Evebox dentro de screen

```
# screen -S evebox
# /usr/bin/evebox -v -D /var/lib/evebox --datastore sqlite --input /var/log/suricata/eve.json
```

Após a inicialização do Evebox deveremos ter a saída abaixo em execução (Figura 46).

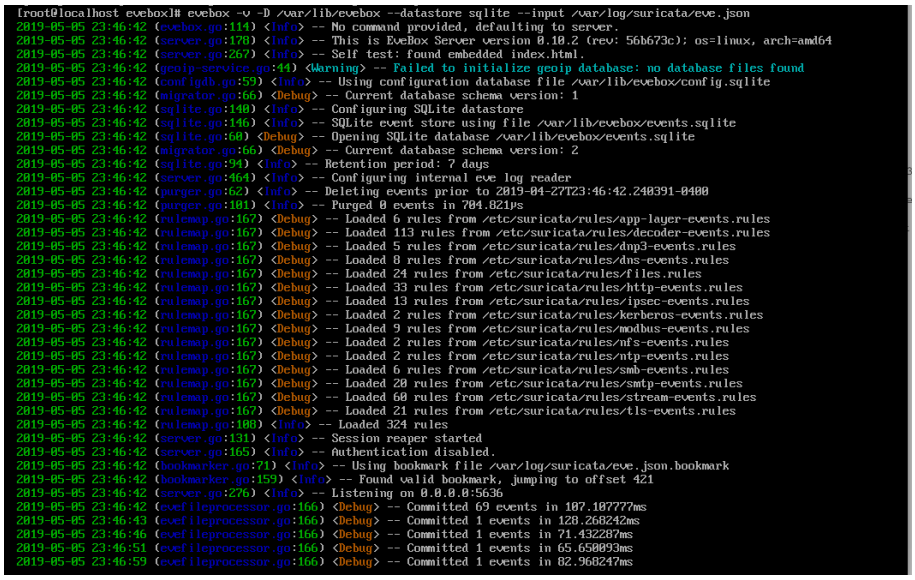

Figura 46 – Evebox em execução

Para sair do *screen* e deixar o Evebox em execução, aperte as teclas ’Ctrl + A’ e depois apenas o ’d’.

Após a execução do Evebox via linha comando é possível acessar pelo *browser* no endereço http://192.168.1.2:5636/ . As telas abaixo (Figuras 47 e 48) mostram o sistema e principalmente na Figura 48, dados coletados pelo Suricata IDS, carregados e mostrados via Evebox. A primeira tela (Figura 47) apresenta informações apenas quando há alertas pelo Suricata IDS.

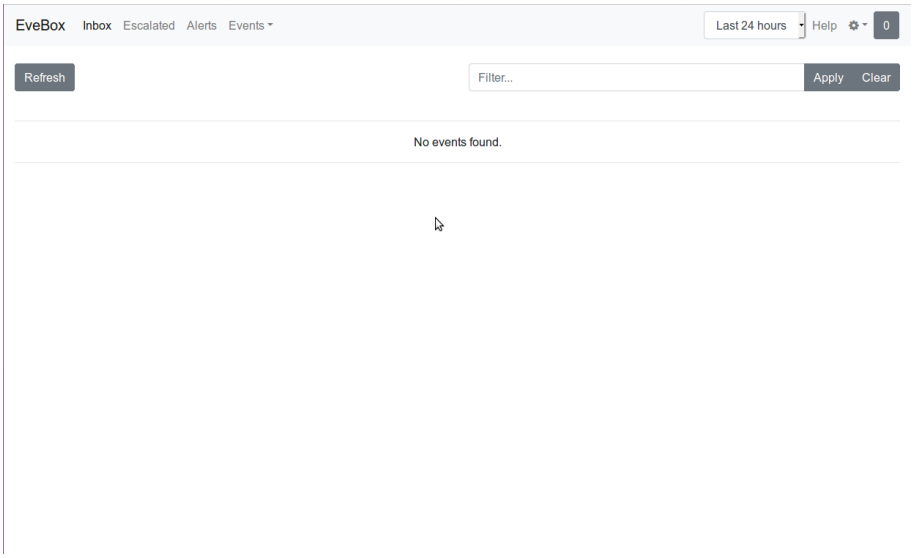

Figura 47 – Dados do Evebox - 01

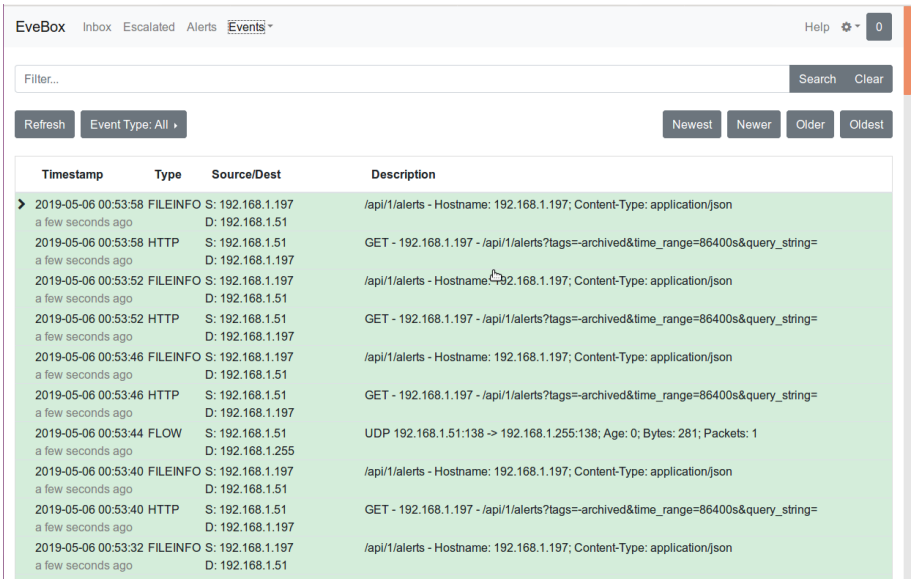

Figura 48 – Dados do Evebox - 02

A.1.3 Virtualização

Para configuração do ambiente de virtualização seguem algumas premissas básicas para o ambiente:

1. Conectividade com a Internet já implementada
2. Servidor instalado do Ambiente de Monitoração

3. Servidor para instalação do Ambiente de Virtualização

Segundo o plano de endereçamento inicial, para rede interna, o servidor para instalação do Ambiente de virtualização terá um fixo e será 192.168.1.3/24.

O Ambiente de Virtualização deve ser instalado em um Servidor com sistema operacional Linux ou Windows. Cada um destes ambientes pode ser utilizado com uma finalidade de virtualização. Para que o ativo de TIC em teste não se considere no mesmo, em nosso caso, utilizamos cada um destes ambientes com os seguintes propósitos:

1. Virtualização de dispositivos para simulação de uma rede real, ou seja, o ativo de TIC pode verificar sua vizinhança e dado que há uma quantidade de outros ativos de TIC, o mesmo não se considera em testes.

2. Virtualização dispositivos em volta para simular acesso ao ativo de TIC e para ambiente de monitoração verificar o tráfego de rede;

3. Virtualização do ativo de TIC em teste, se for possível.

Assim, segue abaixo algumas das configurações utilizadas. É importante lembrar que, no caso dos processadores x86, as tecnologias Intel VT-X e AMD-v orientam o suporte a virtualização. Em todos os casos os servidores tinham 8 GB de memória RAM.

Hardware do Servidor de Virtualização					
#	CPU	Armazenamento	Sistema Operacional	Hypervisor	Propósito
1	Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz	148GB	Ubuntu 16.04.6 LTS	KVM e QEMU	1 e 2
2	Intel(R) Xeon(R) CPU 5140 @2.33GHz	126GB	Kali GNU/Linux Rolling	KVM e QEMU	1 e 2
3	Intel(R) Core(TM) i7-4790 @3.60GHz	500GB	Windows 10	Oracle VirtualBox	1 e 3

Para identificar que seu processador possui suporte a virtualização execute o comando abaixo, como root, se for Linux:

Comando para identificar suporte a virtualização no processador

```
# lscpu | grep Virtualization
```

Oracle VirtualBox

O Oracle VirtualBox é voltado para virtualização de plataforma x86 e AMD64/Intel64. Pode ser instalado em diversos sistemas operacionais. A lista dos sistemas operacionais suportados e o download pode ser feito em *https://www.virtualbox.org/wiki/Downloads*.

Para instalar o Oracle VirtualBox, no Linux, deve-se saber, previamente, se sua distribuição é suportada. Abaixo exemplos para sistemas baseados em pacotes DEB e RPM. Sugerimos distribuição Linux baseada em RPM o CentOS 7 e como distribuição Linux baseada em DEB o Ubuntu 16.

Para sistemas baseados em pacotes DEB executar:

Exemplo de instalação do Oracle Virtualbox - DEB

```
$ wget -q https://www.virtualbox.org/download/oracle_vbox_2016.asc -O- | sudo apt-key add -
$ wget -q https://www.virtualbox.org/download/oracle_vbox.asc -O- | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install virtualbox-6.0
```

Para sistemas baseados em pacotes RPM, como root, executar:

Exemplo de instalação do Oracle Virtualbox - RPM

```
# cd /etc/yum.repos.d
# wget http://download.virtualbox.org/virtualbox/rpm/rhel/virtualbox.repo
# yum install VirtualBox-6.0
```

Exemplos de criação de máquinas virtuais para o Oracle Virtualbox podem ser obtidos em livro, como em ”Getting started with oracle vm virtualbox”[Dash, 2013], além de tutoriais na internet nesta lista:

- Android - https://github.com/carlos-teles/etsg/blob/master/vm/oracle-vm-android.pdf
- Ubuntu - https://github.com/carlos-teles/etsg/blob/master/vm/oracle-vm-ubuntu.pdf
- Windows 10 - https://github.com/carlos-teles/etsg/blob/master/vm/oracle-vm-windows10.pdf

KVM e QEMU

O KVM e o QEMU funcionam em conjunto para virtualização em sistema Linux. Para instalação em sistemas baseados em pacotes RPM, como root, executar:

Exemplo de instalação do KVM e QEMU - RPM

```
# yum install qemu-kvm libvirt libvirt-python libguestfs-tools virt-install qemu-system*
```

Após a instalação, habilite o serviço *libvirtd* e inicie-o com os comandos abaixo como root:

Habilitando serviço libvirtd

```
# systemctl enable libvirtd
# systemctl start libvirtd
```

Para instalação em sistemas baseados em pacotes RPM, como root, executar:

Exemplo de instalação do KVM e QEMU - DEB

```
# apt install qemu-system* qemu-kvm libvirt-bin virtinst bridge-utils cpu-checker
```

Os serviços, como o libvirtd, são habilitados automaticamente.

Para o KVM e QEMU existem diversas formas de se criar máquinas virtuais. Em nosso exemplo, uma das necessidades, era iniciar várias máquinas virtuais dentro da rede de testes dos ativos de TIC. Para esta solução devemos utilizar o conceito de *bridge* (ponte).

Quando iniciamos o KVM e QEMU sem nenhum parâmetro referente a rede, ele inicia com uma configuração padrão, onde se criará uma rede, em que as máquinas virtuais acessam o *host*, a rede e inclusive a internet (caso o *host* também acesse). Mas não acessam as outras máquinas virtuais e nem estarão acessíveis por outras máquinas da rede, ou seja, as máquinas virtuais não se veem na rede e nem os ativos de TIC conectados. Isto fica aquém de nossa necessidades, pois, precisamos criar um ambiente que proporcione algo o mais próximo de um ambiente de rede para que o ativo de TIC em teste não realize que se encontre no mesmo.

Sendo assim, é melhor fazermos uma configuração mais realista, onde a máquina virtual se conecte na rede física, obtendo um IPv4 ou IPv6 do DHCP e se comporte como uma máquina real, além disto proporcionar um melhor desempenho.

Para isso usaremos a configuração TAP e criaremos uma bridge para que as máquinas virtuais se conectem à rede através desta.

O nome da *bridge* será *br0*.

Para verificar se a mesma não existe, execute o comando abaixo para verificar as configurações de rede:

Verificação de configurações de rede

```
# ifconfig

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.3  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::2413:f8df:9426:632c  prefixlen 64  scopeid 0x20<link>
ether 20:47:47:ab:52:d3  txqueuelen 1000  (Ethernet)
RX packets 3067  bytes 2912233 (2.9 MB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 2229  bytes 780577 (780.5 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 20  memory 0xf7c00000-f7c20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Loopback Local)
RX packets 589  bytes 45834 (45.8 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 589  bytes 45834 (45.8 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Assim, verificamos que não existe a *bridge* br0 criada no sistema. Para verificar se existe alguma *bridge* no sistema, execute o comando abaixo:

Verificação de existência de *bridge*

```
#brctl show
bridge name bridge id   STP enabled interfaces
```

A saída do comando indica que não existe nenhuma *bridge* criada no sistema. Para criar a *bridge* br0 e verificar sua existência, execute o comando abaixo:

Criação e verificação de *bridge*

```
# brctl addbr br0
# brctl show
bridge name bridge id   STP enabled interfaces
br0   8000.000000000000 no
```

Após as execuções acima devemos acrescentar a interface física eno1 a nossa *bridge* br0 e fazer com que a mesma pegue um endereço do DHCP, para isso, executaremos o seguinte:

Configurção e verificação de *bridge*

```
# brctl addif br0 eno1
# brctl show
bridge name bridge id   STP enabled interfaces
br0   8000.204747ab52d3 no    eno1
# dhclient -v br0
# ifconfig
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.101  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::2247:47ff:feab:52d3  prefixlen 64  scopeid 0x20<link>
ether 20:47:47:ab:52:d3  txqueuelen 1000  (Ethernet)
RX packets 361  bytes 43262 (43.2 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 416  bytes 646930 (646.9 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.3  netmask 255.255.255.0  broadcast 192.168.1.255
inet6 fe80::2413:f8df:9426:632c  prefixlen 64  scopeid 0x20<link>
ether 20:47:47:ab:52:d3  txqueuelen 1000  (Ethernet)
RX packets 9388  bytes 9188783 (9.1 MB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 7009  bytes 2615262 (2.6 MB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
device interrupt 20  memory 0xf7c00000-f7c20000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Loopback Local)
RX packets 755  bytes 60521 (60.5 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 755  bytes 60521 (60.5 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Com a configuração acima criada, já a *bridge* criada e a interface física associada a mesma. Esta configuração fará que ao acrescentarmos uma nova interface, neste caso lógica, para as máquinas virtuais na *bridge*, teremos conectividade com rede.

Para criar uma máquina virtual leve e de forma rápida sugerimos a instalação do pacote *arm_now*.

O *arm_now* é uma ferramenta de virtualização que se utiliza como base o QEMU para uma configuração instantânea de máquinas virtuais [nongiach, 2018].

Para instalação *arm_now* siga os seguintes passos:

Instalação do arm_now

```
# mkdir -p /usr/local/armmow
# cd /usr/local/armmow
# virtualenv -p python3.6 arm-now
# source bin/activate
# pip install https://github.com/nongiach/arm_now/archive/master.zip --upgrade
```

Após a instalação, execute o comando abaixo para verificar se o arm_now foi instalado corretamente:

Verifica arm_now e lista plataformas disponíveis

```
# arm_now list
aarch64
armv5-eabi
armv6-eabihf
armv7-eabihf
m68k-coldfire
microblazebe
microblazeel
mips32
mips32el
mips32r5el
mips32r6el
mips64-n32
mips64el-n32
nios2
powerpc64-e5500
powerpc64-power8
powerpc64le-power8
sh-sh4
x86-64-core-i7
x86-core2
x86-i686
xtensa-lx60
```

Com isso, poderemos instanciar máquinas virtuais com as arquiteturas acima, em nossa rede, de forma que tenhamos tantas quantas possíveis em nosso servidor de virtualização. Para cada máquina virtual deve-se criar um TAPX, onde X é a numeração, e associar o mesmo a *bridge*. Assim abaixo o exemplo de criação do TAP1, associação a bridge *br0*, geração de *MAC Address* aleatório e instancia da máquina virtual do tipo armv5-eabi.

Máquina virtual com arm_now

```
# cd /usr/local/armnow/arm-now
# tunctl -t tap1 -u `whoami`
Set 'tap1' persistent and owned by uid 0
# ifconfig tap1
tap1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
ether 6e:ce:3a:fa:08:a1  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# brctl addif br0 tap1
# ip link set up dev tap1
# ifconfig tap1
tap1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
ether 6e:ce:3a:fa:08:a1  txqueuelen 1000  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# brctl show
bridge name bridge id   STP enabled interfaces
br0   8000.204747ab52d3 no    eno1
tap1

# printf 'DE:AD:BE:EF:\%02X:\%02X\n' $((RANDOM\%256)) $((RANDOM\%256))
DE:AD:BE:EF:A9:26
# arm_now start armv5-eabi --add-qemu-options="-net nic,model=lan9118,macaddr=DE:AD:BE:EF:A9:26
   -net tap,ifname=tap1,script=no,downscript=no"
...
press ctrl+] to kill qemu

Welcome to arm_now
buildroot login:
```

Caso apareça o prompt acima, basta digitar 'root' e depois enter e a máquina virtual já está ligada. Entretanto, ainda é necessário que a maquina virtual seja conectada à rede. Na máquina virtual, execute os comandos:

Configuração de rede da máquina virtual com arm_now

```
# route del default gw 10.0.2.2
# ifconfig eth0 192.168.1.64
# route add default gw 192.168.1.1
# vi /etc/resolv.conf
# opkg install isc-dhcp-client-ipv4
# dhclient
# ifconfig
eth0      Link encap:Ethernet  HWaddr DE:AD:BE:EF:A9:26
inet addr:192.168.1.225  Bcast:192.168.1.255  Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:541 errors:0 dropped:2 overruns:0 frame:0
TX packets:405 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:719261 (702.4 KiB)  TX bytes:26729 (26.1 KiB)
Interrupt:31
```

Este procedimento somente é necessário uma vez, pois, no próximo *boot* da máquina virtual a mesma virá configurada corretamente.